COSC 2206 Internet Tools

JavaScript

Client-side Scripting

Document Object Model

Client-side JavaScript (1)

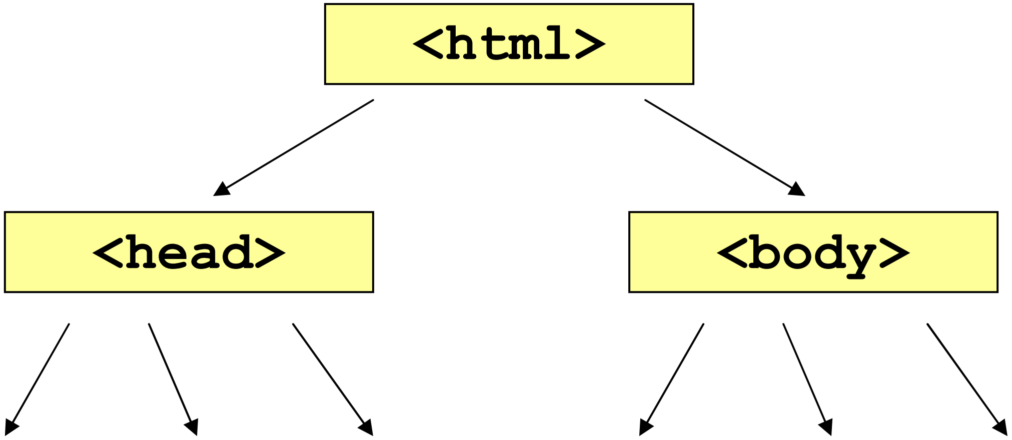
- JavaScript can be used as a standalone scripting language not associated with a browser.
- Microsoft allows this with JScript. It can be used with the Windows Scripting Host (WSH) as a scripting language (replacement for batch files)
- The terminology "client-side" refers to JavaScript and its interaction with the browser DOM (Document Object Model).

Client-side JavaScript (2)

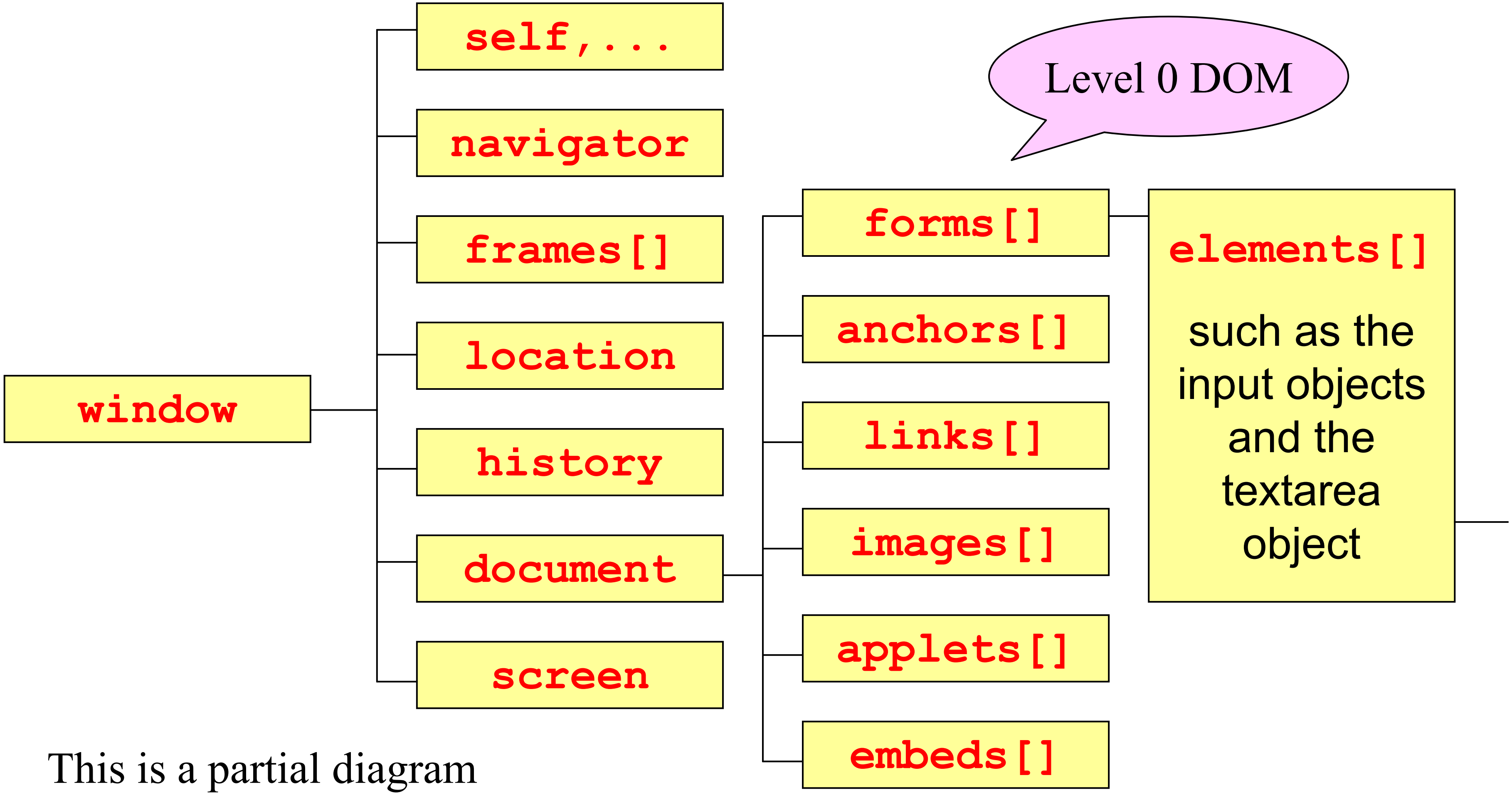
- A browser defines objects such as the top level **`window`** object and the **`document`** object that can interact with JavaScript.
- This object hierarchy provides access to the document object model (DOM) which is being standardized by the W3C.
- There is also an event model that provides user interaction by associating events with JavaScript methods.

Client-side JavaScript (3)

- The combination of JavaScript, DOM, and CSS is often called Dynamic HTML
- In the DOM a web page has a tree structure



Client-side object hierarchy



This is a partial diagram

DOM0 properties

- This example shows how to examine the DOM 0 properties of an object

`examples/dom/dom0.html`

Alert, Confirm, Prompt (1)

- Interaction with user using dialog boxes
- `window.alert(msg);`
  - displays a message in a dialog box that is dismissed by clicking OK.
- `var r = window.confirm(msg);`
  - displays a message (question and asks for confirmation (OK and Cancel buttons). Returns true if user clicks OK, else returns false

Alert, Confirm, Prompt (2)

- `var r = window.prompt(msg,default);`
  - displays a message and a text box. The user enters some input into the text box and clicks OK, Clear, or Cancel button.
  - The string entered by the user is the return value. If no input is entered an empty string is returned.
  - If the Cancel button is clicked `null` is returned

Alert, Confirm, Prompt (3)

```
alert(
"This program can add two numbers\n" +
"You will be asked to enter two numbers\n" +
"Then you will be asked if you want to add them\n"
);
var first, second, num1, num2, sum;
firstNumber = window.prompt(
   "Enter 1st integer to add", "0");
document.write("<br>", "You entered " + first);
second = window.prompt(
   "Enter 2nd integer to add", "0");
 document.write("<br>", "You entered " + second);
```

Alert, Confirm, Prompt (4)

```
var ok = confirm(
   "Do you want to add these two numbers?");
if (ok)
{
   num1 = parseInt(first);
   num2 = parseInt(second);
   sum = num1 + num2;
   document.write("<br>", "The sum of " +
      first + " and " + second + " is " + sum);
}
```

examples/simple/dialog.html

Getting input from forms (1)

```
<form name="myForm">
   <input name="myInput" type="text"
      value="">
   ...
</form>
```

- JavaScript can refer to the form using
  - `document.myForm`
- The input element value can be referenced using
  - `document.myForm.myInput.value`

Getting input from forms (2)

```
<form name="myForm">
   ...
   <input name="myButton" type="button"
      value="clickMe"
      onclick="getInput()">
</form>
```

- When the button is clicked `getInput` is called and it refers to elements in the form using
  - `document.myForm.elementName`

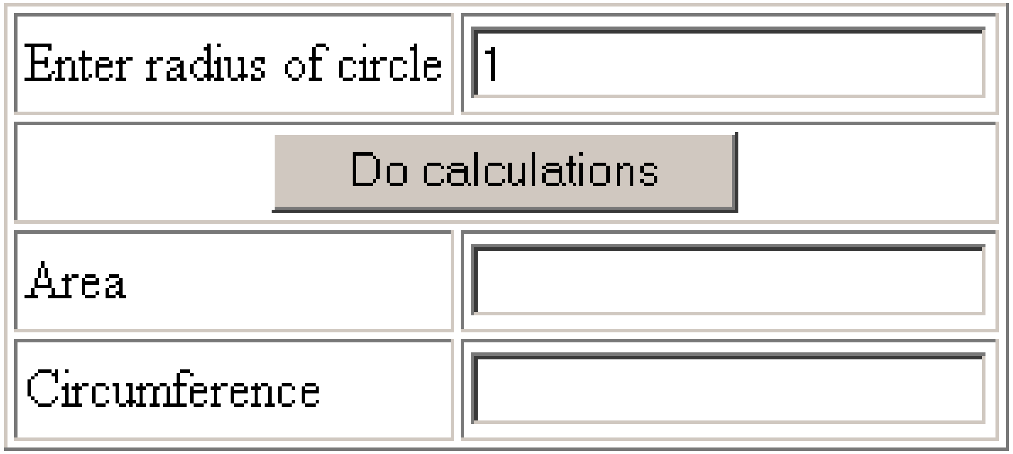
Getting input from forms (3)

```
<form name="myForm">
...
<input name="myButton" type="button"
     value="clickMe"
     onclick="getInput(myForm)">
</form>
```

- When the button is clicked `getInput` is called. Here we use the form name as an argument. It is same as `window.document.myForm`

circleCalculations (1)

- Write a program that calculates the area and circumference of a circle:



Enter radius of circle	1
Area	
Circumference	

circleCalculations (2)

use table to line up elements

```
<form name="myForm">
...
<input name="radiusField" type="text" value="1">
...
<input name="calculate" type="button"
   value="Do calculations"
   onclick="displayResults(document.myForm)">
...
<input name="outputAreaField" type="text"
   readonly="true">
<input name="outputCircumferenceField"
   type="text" readonly="true">
</form>
```

circleCalculations (3)

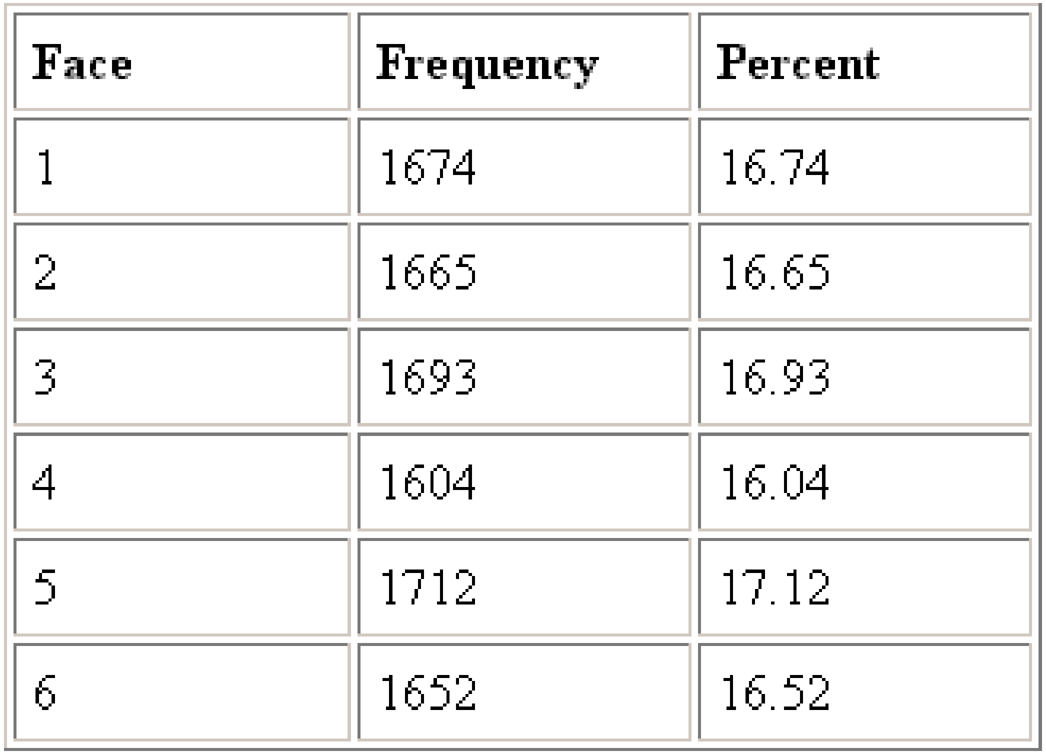
```
function area(r)
{ return Math.PI * r * r; }
function circumference(r)
{ return 2.0 * Math.PI * r; }

function displayResults(form)
{   var r = parseFloat(form.radiusField.value);
    form.outputAreaField.value = area(r);
    form.outputCircumferenceField.value =
       circumference(r);
}
```

examples/apps/circleCalculations.html

Die frequency simulation (1)

- Use JavaScript to produce a frequency table for simuating the rolling of a die:



Face	Frequency	Percent
1	1674	16.74
2	1665	16.65
3	1693	16.93
4	1604	16.04
5	1712	17.12
6	1652	16.52

Die frequency simulation (2)

```
var frequency = new Array(6);
var faceValue; // 1 to 6
var trials; // number of throws in simulation
trials = parseInt(window.prompt(
    "How many trials do you want?", "6000"));

// Initialize frequencies to 0

for (var k = 0; k < frequency.length; k++)
{
    frequency[k] = 0;
}
```

Die frequency simulation (3)

```
// Calculate frequency table

for (var roll = 1; roll <= trials; roll++)
{
    faceValue = Math.floor(1 + Math.random()*6);
    frequency[faceValue - 1]++;
}
```

Die frequency simulation (4)

```
document.writeln(
    "<h1>Frequency table for " + trials +
    " rolls of a die</h1>");
document.writeln(
    "<table border='1' cellpadding='5'
    width='50%'>");
document.writeln("<tr>");
document.writeln("<th align='left'
   width='33%'>Face</th>");
document.writeln("<th align='left'
   width='33%'>Frequency</th>");
document.writeln("<th align='left'
   width='33%'>Percent</th>");
document.writeln("<\tt>");
```

Die frequency simulation (5)

```
for (var k = 0; k < frequency.length; k++)
{   var percent = 100*(frequency[k]/trials);
    percent = Math.round(100*percent)/100;
    document.writeln("<tr><td>" + (k+1) +"</td>");
    document.writeln("<td>" + frequency[k] + "</td>");
    document.writeln("<td>" + percent + "</td>");
    document.writeln("</tr>");
}
document.writeln("</table>");
```

examples/apps/dieSimulation1.html

Die frequency simulation (6)

- Can also write a version that displays results in a separate window using

```
var w = window.open("","result",
 "resizable,menubar,width=400,height=
 400");
```

- Then use `w.document.writeln(...)`

examples/apps/dieSimulation2.html

The game of craps

- A game with two dice
- Player rolls two dice each round
- The `onload` load event handler is used to initialize the game

`examples/apps/crapsGame.html`

Bar graph using 1 by 1 gif

- Shows how to use DOM to manipulate image height and width properties

`examples/apps/barGraph.html`

Opening windows (1)

- Syntax

```
var w = window.open(URL, windowName);
var w = window.open(URL, windowName,
  windowFeatures);
```

- *URL* specifies what document to display
- *windowName* is used in the target attribute of a frame or tag to refer to the window
- *windowFeatures* is a string of window properties

Opening windows (2)

- Some window features
- **`height,width`** window size
- **`resizable=yes`** allow resizing of window
- **`menubar=yes`** to create the menu bar
- **`scrollbars=yes`** to use scroll bars
- **`titlebar=yes`** window has a title bar
- **`toolbar=yes`** window has a tool bar
- **`status=yes`** status bar at window bottom

Opening windows (3)

No spaces!

- Example:

```
var popUp = window.open("", "output",
  "width=200,height=200,menubar=yes,re
  sizable=yes,toolbar=yes,scrollbars=y
  es,status=yes");
```

- The URL is absent here because this is a window that will be written to by JavaScript using statements such as
- `popUp.document.write("...");`

Opening windows (4)

- Clicking a link to open a new window:

```
<a href="#" onclick="openPopUp('popUp.html')">
  Click to open</a>
```

- Here "#" means no link is actually taken.

```
function openPopUp(url)
{
   popUp = window.open(url,"popUp",
         "width=200,height=200");
   popUp.focus();
}
```

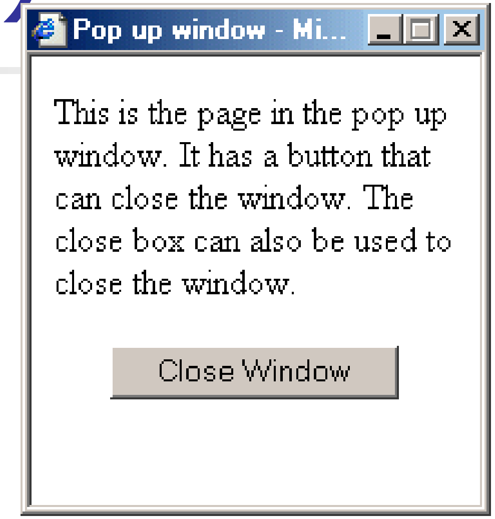
Opening windows (5)

- popUp.html

```
...
<body>
This is the page in the pop up window.
It has a button that can close the window.
<p><form>
<center>
<input type="button" value="Close Window"
   onClick="window.close()">
</center>
</form>
</p></body>
```

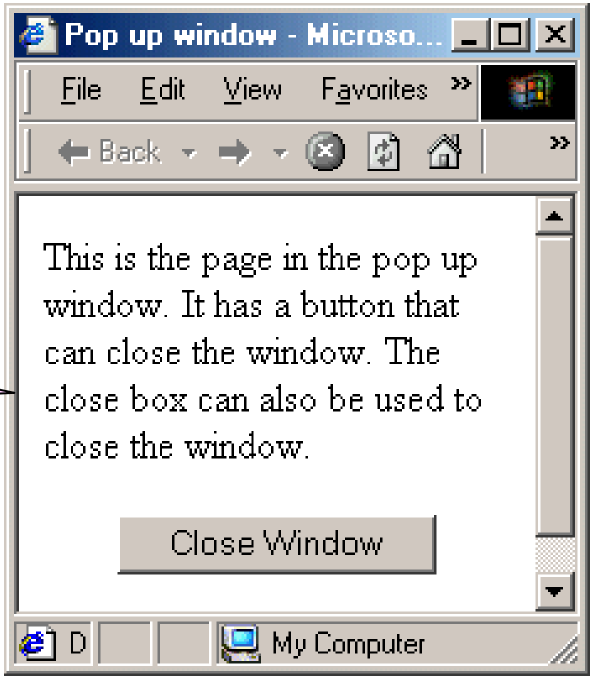
Opening windows (6)

This is the window with default values for the window features



This is the page in the pop up window. It has a button that can close the window. The close box can also be used to close the window.

This is the window with yes values for the window features



This is the page in the pop up window. It has a button that can close the window. The close box can also be used to close the window.

Opening windows (7)

- Clicking a link to open a new window:

```
<a
  href="javascript:openPopUp('popUp.html')">
  Click to open</a>
```

- This is a special JavaScript link

```
function openPopUp(url)
{
   popUp = window.open(url,"popUp",
         "width=200,height=200");
   popUp.focus();
}
```

Opening windows (8)

- HTML document that illustratrates window opening

`examples/windows/windows.html`

A simple back button

- There is a history object so a link can be used:

```
<a href="#"
   onclick="history.go(-1)">Back</a>
```

- Alternatively, a button can be used:

```
<input type="button" value="Back"
  onclick="history.go(-1)">
```

Image and window control

- JavaScript and DOM can be used to control images and windows using links.
- Examples
  - Using a link to replace an image with another one
  - A simpler version using the images[ ] array
  - Using links to trigger events
  - A pop up window with links that display in the parent window.

Changing and image (1)

- The `onclick` event handler can be used to replace one image by another when the link is clicked.
- Give the `<img>` tag a name:

```
<img name="myName" src="pic1.gif">
```

- The image can be replaced by modifying the src attribute using the JavaScript statement

```
document.myName.src = "pic2.gif";
```

Changing an image (2)

- HTML document that displays a red arrow and a blue arrow and contains links that change the arrows when a link is clicked.

examples/links/linkImages1.html

Changing an image (3)

- Images should be preloaded:

```
var redArrow = new Image();
redArrow.src = "redArrow.gif";
var blueArrow = new Image();
blueArrow.src = "blueArrow.gif";
```

Changing an image (4)

- The two arrows are initially specified by the **<img>** tags

```
<img name="topArrow"
    src="redArrow.gif">
<img name="bottomArrow"
   src="blueArrow.gif">
```

- They can be modified using links like

```
<a href="#"
   onclick="changeTopToBlue()">
   Change to blue</a>
```

could also use javascript:

Changing an image (5)

- Functions can be uses to change the images

```
function changeTopToBlue()
{
   window.document.topArrow.src = blueArrow.src;
}
function changeTopToRed()
{
   window.document.topArrow.src = redArrow.src;
}
// similar pair of functions for bottomArrow
```

Changing an image (6)

- When several images are involved it is better to use the **images[]** array (DOM 0)
- Only one function is needed:

```
function changeTo(imgName,objectName)
{
   document.images[imgName].src =
      eval(objectName + ".src");
}
```

eval is necessary

Changing an image (7)

- Now the links can be expressed as

```
<a href="#"
  onclick="changeTo('topArrow',
  'blueArrow')">change to blue</a>
```

- The single quotes are necessary

examples/links/linkImages2.html

Links as event triggers (1)

- The onClick handler can also be used in other ways.
- to confirm if a link should be taken
- to dynamically modify the document
  - Example: changing a color

Links as event triggers (2)

- An onclick handler that asks for confirmation

```
function confirmLink()
{
    return confirm(
    "Do you want to follow this link");
}
```

- This can be used as follows

```
<a href="destination.html" onclick="return
  confirmLink()">
  destination</a>
```

Links as event triggers (3)

- `<a href="destination.html" onclick="return confirmLink()">destination</a>`
- The return is necessary here because when you write something like `onclick="..."` the quotes delimit the body of a fictitious JavaScript wrapper function associated with the event handler.

Links as event triggers (4)

- The following function can be used to dynamically change the document colors

```
function changeColor()
{
    var bColor = prompt("...");
    var fColor = prompt("...");
    document.bgColor = bColor;
    document.fgColor = fColor;
}
```

Links as event triggers (5)

- Use the following link to change colors

```
<a href="#" onclick="changeColor()">
  Click to change colors</a>
```

examples/links/linkEvents.html

Pop up links window (1)

- Display a pop up window with links whose documents are displayed in the main window.
- The `popUpLinks.html` file contains the function

```
function showPage(url)
{   window.opener.location = url;
    window.focus();
}
```

whoever opened the pop up window

Pop up links window (2)

- The links are specified as

```
<a href="#"
  onclick="showPage('....');">
  ...</a>
```

- The main page contains the statement

```
var w = window.open("popUpLinks.html",
  "links", "width=100,height=200");
```

examples/links/popUpLinksMain.html

Slide show (1)

- Use JavaScript to set up a slide show with navigation buttons.

`examples/slideShow/slideShow.html`

Slide show (2)

- HTML part uses the `<img>` tag:

```
<img src="images/weblab01.jpg"
  name="picture">
```

- The links are done using

```
<a href="javascript:start()">Start</a>
<a href="javascript:nextPicture(-1)">
Previous</a>
<a href="javascript:nextPicture(1)">
Next</a>
```

Slide show (3)

- Put the image files in an array:

```
var pics = new Array(
    "weblab01", "weblab02", ...
    "weblab17");
```

- Make file names

```
for (var k = 0; k < pics.length; k++)
   pics[k] = "images/" + pics[k] +
      ".jpg";
```

Slide show (4)

- Global variables

```
var currentIndex = 0;
var maxIndex = pics.length - 1;
```

- Display first picture

```
function start()
{
   currentIndex = 0;
   document.picture.src =
      pics[currentIndex];
}
```

Slide show (5)

- Display next picture using circular index

```
function nextPicture(direction)
{   currentIndex += direction;
    if (currentIndex > maxIndex)
        currentIndex = 0;
    else if (currentIndex < 0)
        currentIndex = maxIndex;
    document.picture.src =
        pics[currentIndex];
}
```

Rollover (1)

- A simple rollover is a pair of images which are alternated when the mouse moves over them
- More advanced rollovers can simulate the pushing and releasing of a button using rollover images and images for mouse down and up.

Rollover (2)

- An arrow that changes color as the mouse is moved over it.
- This is done with a red arrow image and a blue arrow image

examples/rollOver/simpleRollover1.html

Rollover (3)

- First pre-load the images

```
var overArrow = new Image();
overArrow.src = "redArrow.gif";
var defaultArrow = new Image();
defaultArrow.src = "blueArrow.gif";
```

Rollover (4)

- Function to change an image

```
function changeTo(imgName, objectName)
{
   document[imgName].src =
      objectName.src;
}
```

Rollover (5)

- HTML to do the rollover

```
<a href="newPage.html"
onmouseover="changeTo('arrow',
   overArrow)"
onmouseout="changeTo('arrow',
   defaultArrow)">
<img name="arrow" src="blueArrow.gif"
   border="0"</a>
```

The latest browsers allow events inside img tag so there is no need to wrap it inside a link

Rollover (6)

- With the latest browsers the link is not needed:

```
<img name="arrow" src="blueArrow.gif"
  border="0" onclick=
  "window.location='newPage.html'"
   onmouseover=
   "changeTo('arrow',overArrow)"
   onmouseout=
   "changeTo('arrow',defaultArrow)">
```

examples/rollover/simpleRollover2.html

Other rollover examples

examples/rollOver/rollover1.html

examples/rollOver/rollover2.html

examples/rollOver/rollover3.html

examples/rollOver/rollover4.html

A multiple choice quiz

examples/frames/quizFrames.html

examples/frames/page1.html

examples/frames/page2.html

examples/frames/statistics.html

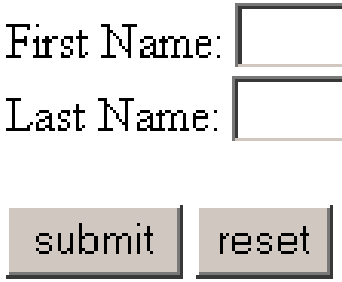
examples/frames/statistics.js

Form Validation

- JavaScript can be used to verify a form before the form data is sent to the server-side script for processing.
- This reduces the number of connections to the server in case of incorrect or missing data.
- For security reasons form verification should also be done on the server-side.

Form validation example (1)

- Write a form containing a first name and a last name field.
- Check using JavaScript that a first name and a last name were entered



when the submit button is clicked a JavaScript method will be called

Form validation example (2)

```
<p>
<form name="theForm"
  action="/cgi-bin/formPost.pl" method="post"
  onsubmit="return checkForm(theForm.first,
  theForm.last)">
First Name: <input type="text" name="first"><br>
Last Name: <input type="text" name="last"><br>
</p><p>
<input type="submit" value="submit">
<input type="reset" value="reset">
</p>
</form>
```

Form will be submitted only if `checkForm` returns true

Form validation example (3)

```
function checkForm(first, last)
{  var error = "";
   if (first.value == "")
      error += "You must enter a first name";
   if (last.value == "")
      error += "\nYou must enter a last name";
   if (error == "")
      return confirm("No errors found, ...");
   else
   {  alert("The following errors were found\n" +
         error);
      return false;
   }
}
```

Form validation example (4)

- Local version

```
examples/forms/formValidate.html
```

- Server version

```
http://localhost/formValidate.html
```

Email validation example (1)

```
<form name="theForm"
  action="/cgi-bin/formPost.pl" method="post"
  onsubmit="return checkForm(theForm.email)">
<p>
email: <input type="text" name="first">
</p>
<p>
<input type="submit" value="submit">
<input type="reset" value="reset">
</p>
</form>
```

Form will be submitted only if `checkForm` returns true

Email validation example (2)

```
function checkForm(email)
{
    if (! isValidEmailAddress(email.value))
    {
        alert("Email address is invalid");
        email.focus();
        email.select();
        return false;
    }
    return true;
}
```

this is a complicated function

keyboard focus

select the text

examples/forms/emailValidate.html

Email validation example (3)

```
function isValidEmailAddress(emailAddress)
{
   /* Check for empty address or invalid chars */

   if (emailAddress == "" ||
      hasInvalidChar(emailAddress))
   {
       return false;
   }
```

Email validation example (4)

```
/* Check for @ character */

var atPos = emailAddress.indexOf("@", 1);
if (atPos == -1)
{
   return false;
}
```

Email validation example (5)

```
/* Check for @ character */

var atPos = emailAddress.indexOf("@", 1);
if (atPos == -1)
{
   return false;
}
```

Email validation example (6)

```
/* Check that there are no more @ chars */

if (emailAddress.indexOf("@", atPos + 1) > -1)
{
    return false;
}
```

Email validation example (7)

```
/* Check for a dot somewhere after @ */

var dotPos = emailAddress.indexOf(".",
    atPos + 1)
if (dotPos == -1)
{
    return false;
}
```

Email validation example (8)

```
    /* Check for two or more characters after
       the last dot */

    var lastDotPos =emailAddress.lastIndexOf(".");

    if (lastDotPos + 3 > emailAddress.length)
    {
       return false;
    }
    return true;
}
```

Email validation example (9)

```
function hasInvalidChar(emailAddress)
{
   var invalidChars = "/;:,"; // not complete
   for (var k = 0; k < invalidChars.length;
      k++)
   {
      var ch = invalidChars.charAt(k);
      if (emailAddress.indexOf(ch) > -1)
         return true;
   }
   return true;
}
```

Regular Expressions

- A regular expression is a pattern that describes a class of strings.
- Patterns are used to test if a target string contains a match for the pattern
- They are also used to replace one or all occurrences of the pattern with a substring
- The patterns are described by a grammar
- In the simplest case a pattern is just a fixed string such as the word "hello"

Special matching chars (1)

\	escape character
^	match beginning of string
$	match end of string
*	zero or more times
+	one or more times
?	zero or one time (optional)
.	any character except newline

Special matching chars (2)

`\b`	word boundary
`\B`	Non-word boundary
`\d`	any digit 0 to 9(same as [0-9])
`\D`	any non-digit
`\f`	form feed
`\n`	newline
`\r`	carriage return
`\s`	any single whitespace character

Special matching chars (3)

\S	any single non-whitespace char
\t	tab
\v	vertical tab
\w	any letter,number,underscore
\W	any non-letter,-number,-underscore
[abcd]	one of the characters inside [ ]
[^abc]	char other than ones inside [ ]
[a-e]	any char in specified range

Special matching chars (4)

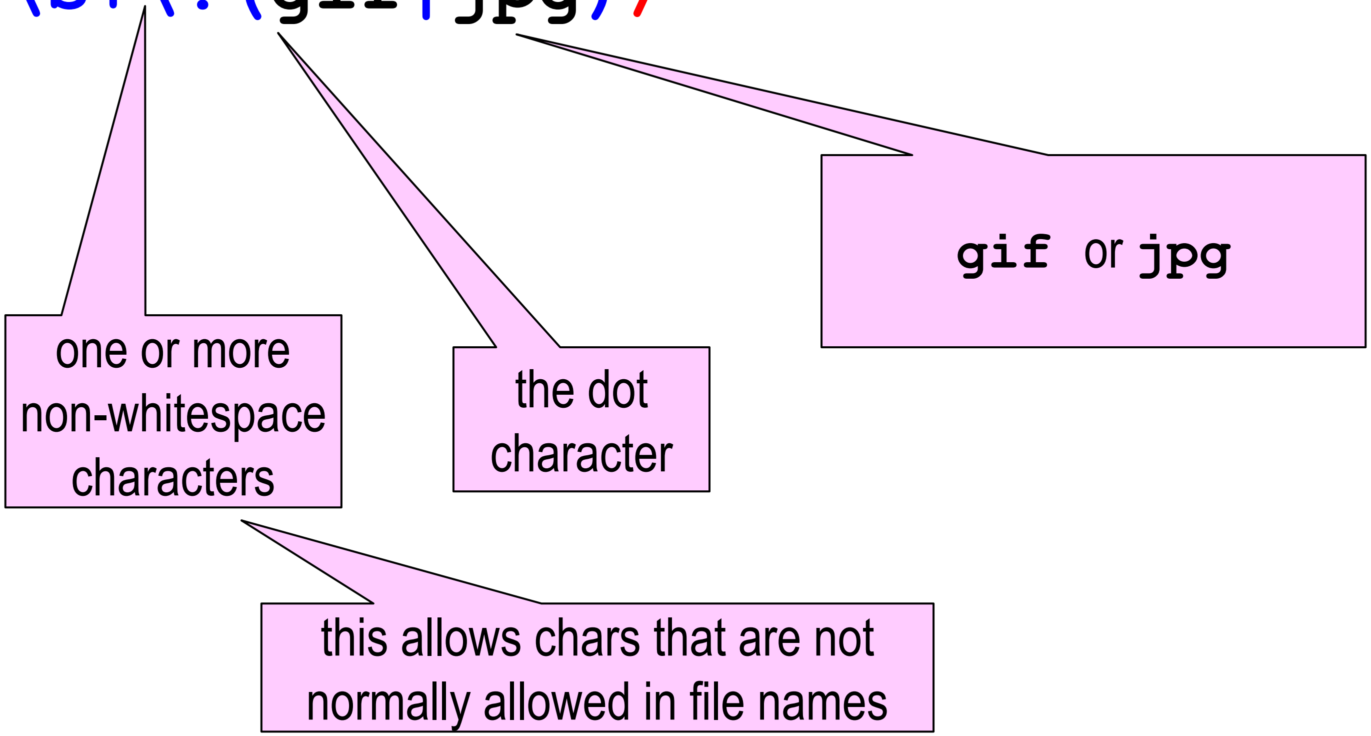
`{n}`	`n` occurrences of previous char
`{,n}`	at least `n` occurrences
`{n,m}`	between `n` and `m` occurrences
`()`	stored grouping
`x|y`	alternation: either `x` or `y`
`[a-e]`	any char in specified range

Examples (1)

- Literal regular expressions patterns are delimited by `/` characters as in `/pattern/`
- Example: `/help/` matches **`help`** anywhere in a string
- Example: `/^help/` matches help only at the beginning of a string
- Example: `/^help$/` matches the string **`help`**
- Example: `/\bhelp\b/` matches help only as a complete word

Examples (2)

- Example: match file names ending in the **`gif`** or **`jpg`** extensions:
  - `/\S+\.(gif|jpg)/`



one or more non-whitespace characters

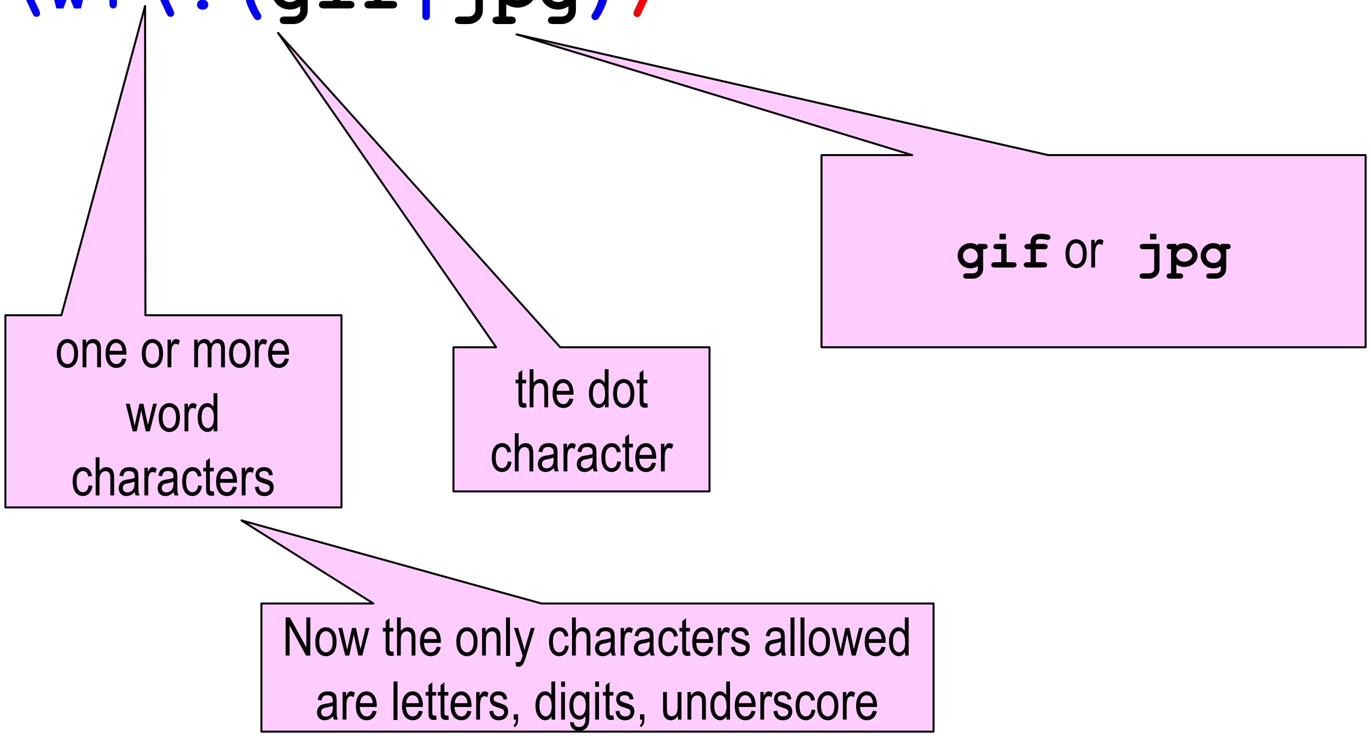
the dot character

`gif` or **`jpg`**

this allows chars that are not normally allowed in file names

Examples (3)

- Example: match file names ending in the **`gif`** or **`jpg`** extensions:
  - `/\w+\.(gif|jpg)/`



one or more word characters

the dot character

`gif` or **`jpg`**

Now the only characters allowed are letters, digits, underscore

Global / Case insensitive

- To match all occurrences of a pattern in a string use the global modifier (useful for replacing all occurrences of a pattern)
  - `/pattern/g`
- To do case insensitive matches use the insensitvie modifier
  - `/pattern/i`
- To do both use
  - `/pattern/gi`

Creating reg exp objects

- Use a literal **`RegExp`** object:
  - `var re = /pattern/attributes;`
  - Example:

    ```
    var re = /^hello$/gi;
    ```
- Use a constructor
  - `var re = new RegExp(pattern, attributes);`
  - Example:

    ```
    var re = new RegExp("^hello$","gi");
    ```

Testing for a pattern

- If regex is a regular expression object and s is a string to search then we can test if the pattern is present in s using the if statement

```
if (regex.test(s))
{
    // pattern was found
}
```

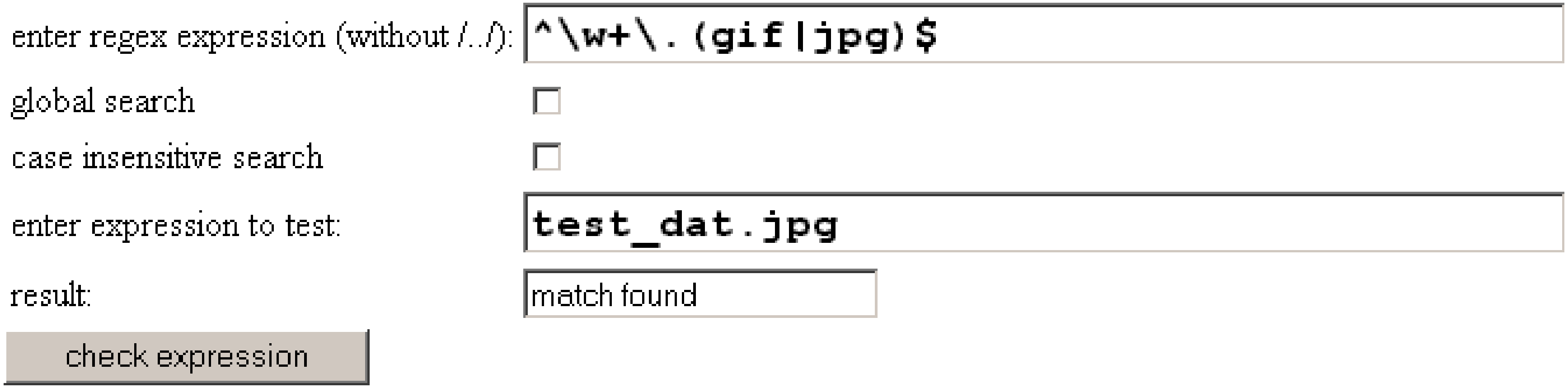
Replacing a pattern

- If **`regex`** is a regular expression object and **`s`** is a string to search and **`replace`** is the string to use to replace occurrences of the pattern then the replacement can be done using

```
var s1 = s.replace(regex,
   replace);
```

Testing a regular expression

- The following html document can be used to test your regular expressions:

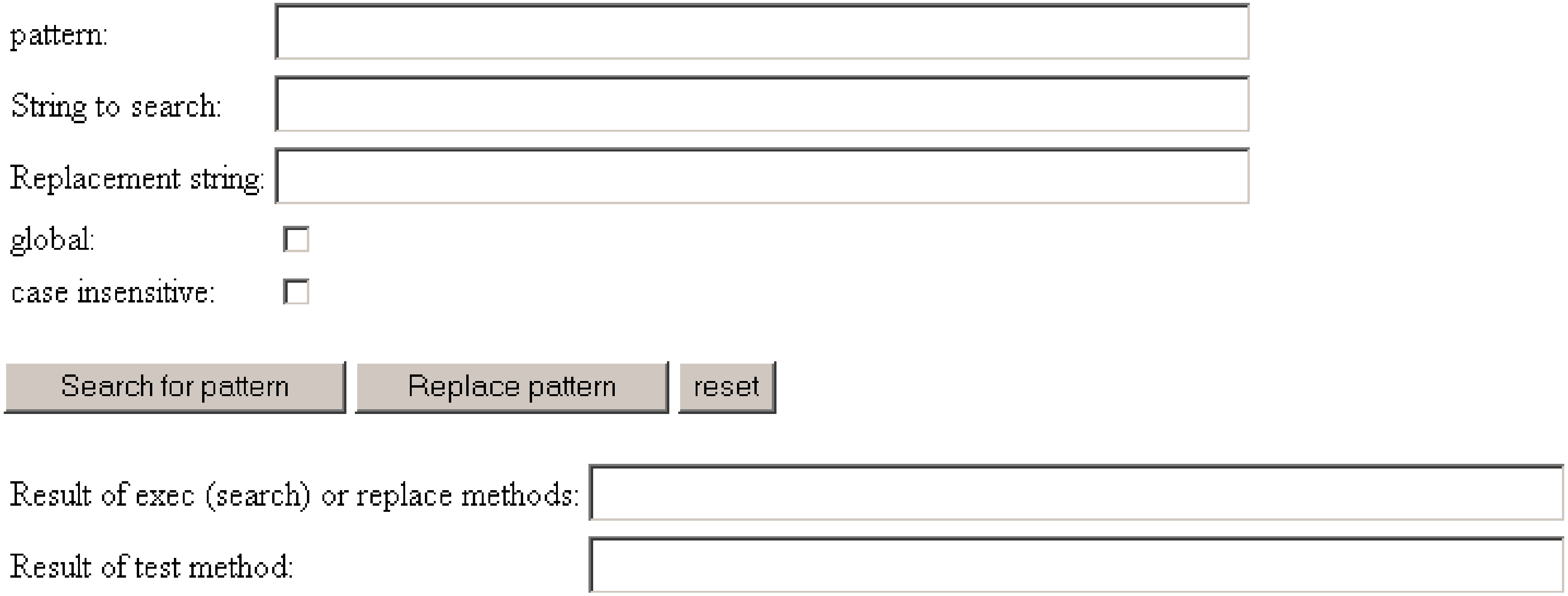


enter regex expression (without /../):	^\w+\.(gif|jpg)$
global search	□
case insensitive search	□
enter expression to test:	test_dat.jpg
result:	match found

examples/forms/testRegExp.html

Search and replace

- Here is a more general tester



pattern:

String to search:

Replacement string:

global: ☐

case insensitive: ☐

Result of exec (search) or replace methods:

Result of test method:

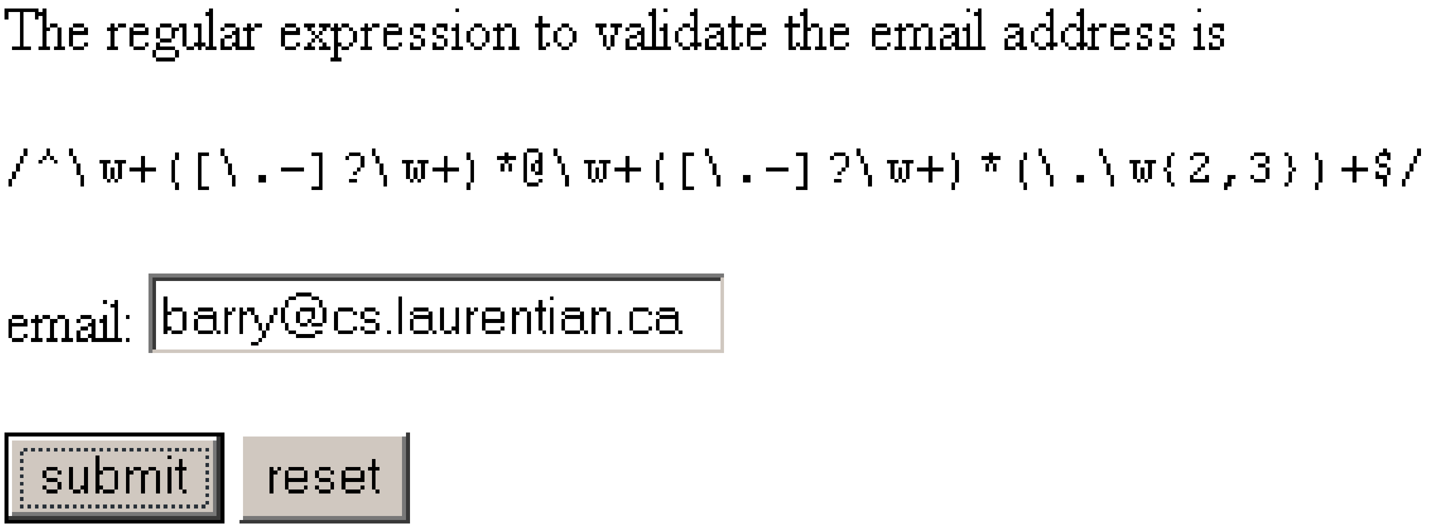
examples/forms/replaceRegExp.html

Validating email address (1)

`/^\w+`	one or more word chars
`([\.-]?\w+)*`	zero or more ocurrences of .word or -word
`@`	the literal @ symbol
`\w+`	word
`([\.-]?\w+)*`	as above
`(\.\w{2,3})+`	ends with 2 or 3 chars
`$/`	

Validating email address (2)

- Email form validation example



The regular expression to validate the email address is

```
/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/
```

email: barry@cs.laurentian.ca

examples/forms/emailValidateRegExp.html

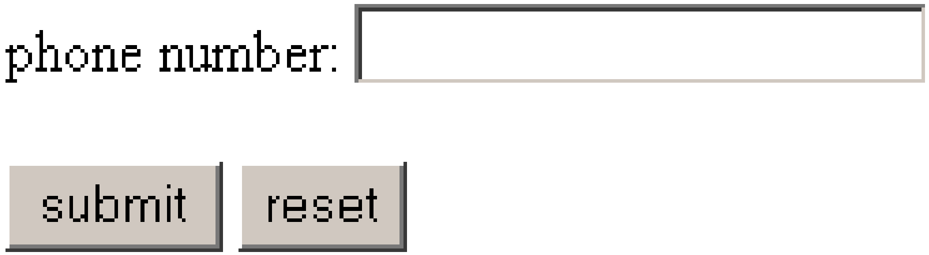
Phone number validation (1)

- Suppose you want to accept phone numbers only from area codes 705, 911, or 416 using hypens as separators
- The following regular expression can be used

```
/(705|911|416)-\d\d\d-\d\d\d\d/
```

Phone number validation (2)

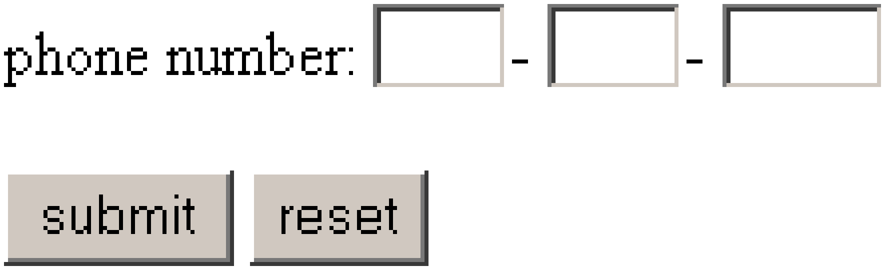
- First method



examples/forms/validatePhoneNumber.html

Phone number validation (3)

- Better interface


phone number: [ ] - [ ] - [ ]

examples/forms/validatePhoneNumber2.html

Accessing the DOM (1)

- The `getElementById` method can be used to access any element:
- The element

```
<div id="myTitle"...>Hello there</div>
```

can be accessed using

```
var obj =
document.getElementById("myTitle");
```

Accessing the DOM (2)

- Each CSS property has a corresponding property in the DOM
- To change the color of an element use

```
obj.style.color = "yellow";
```

- To change the background color use

```
obj.style.backgroundColor ="red";
```

examples/dom/colorChange.html

Accessing the DOM (3)

- Modifying the CSS display property to toggle visibility of blocks of text.
- The display property can be block or none

`examples/dom/toggleText.html`

- Exercise: Do a similar example with two levels of links

Accessing the DOM (4)

- Drop down menus using the CSS visibility property which can have the values visible or hidden. This is the way to implement layers using the DOM method getElementById

```
layer =
    document.getElementById(layerid);
layer.style.visibility = "hidden";
```

examples/dom/dropDownMenus.html

Accessing the DOM (5)

- Implementing layers using div and z-index

```
layer =
    document.getElementById(layerid);
layer.style.zindex = ...;
```

examples/dom/layers.html

Accessing the DOM (6)

- positioning properties
- Simple examples of div positioning and visibility

examples/dom/position.html

Accessing the DOM (7)

- Using CSS and DOM to do link roll overs

examples/dom/rollover.html